General Instructions

- The assignments will help you learn Verilog as a Hardware Description Language and how hardware circuits can be developed using Verilog and FPGA.
- You have to come to the lab during your lab session to carry out the experiments using the FPGA board. You are expected to write the Verilog modules before coming to the lab. Your TAs will check your results and grade your work during the lab session.
- You have to submit the code artifacts through CANVAS.
- Only one member of each group has to submit the assignment. Please Make sure that there is no duplicate submission from your group.
- The maximum possible deadline extension is 2 days. For the extension for one day after the deadline, there will be a penalty of 10% of the grade. For the second day extension, the penalty is another 20% of the grade. No assignment submission will be accepted after 48 hours of the deadline.
- Sharing of the solution outside the group is strictly prohibited. If found guilty, both the involved groups will get 0 in the assignment.
- Each group member should equally contribute to every assignment and should fully understand the submitted solution. The TAs may ask any group member to explain the solution.

Resources

- **Xilinx Spartan 3E FPGA Starter Board**

  You will use Xilinx Spartan 3E FPGA Starter Board for your Verilog Assignments.

  Here is the manual for the FPGA Kit:

  https://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

- **Xilinx ISE WebPACK Design Software**

  You can use Xilinx ISE WebPACK Design Software for writing and simulating your Verilog programs.

  The software can be downloaded from the following location:

  https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html

  Here are some resources to install the software:

  https://embeddedmicro.com/tutorials/mojo-software-and-updates/installing-ise

  http://dreamrunner.org/blog/2012/09/12/install-xilinx-ise-on-the-ubuntu/

- **Verilog Textbook**

  Pong P. Chu. FPGA Prototyping by Verilog Examples: Xilinx Spartan-3 Version, John Wiley & Sons, Inc., 2008.

  http://www2.dc.ufscar.br/∼marcondes/netfpga/FPGAPrototypingByVerilogExamples.pdf

Assignmnet 1 (10 points)

Lab Session: (during the week of March 06, 2017 - March 10, 2017)
Deadline for submission: 11:59 pm on March 10, 2017

Objectives:

- To learn how to use Xilinx ISE WebPACK design software to develop a Verilog module for a hardware circuit, simulate and test it, and program the FPGA chip to implement the hardware.
- To get familiar with the input-output mechanisms of the Xilinx Spartan 3E FPGA kit: slide switches, push-button switches, LEDs and LCD Screen.

Problems:

1. Blinking a LED (2 points)

Follow the instructions in the following tutorial to blink a LED in your FPGA Board.
https://therobotfix.wordpress.com/2011/06/27/getting-started-with-spartan-3e-fpga-and-verilog/
This tutorial illustrates how to use the ISE Webpack design software and describes all the steps to develop a hardware circuit using FPGA.

References:
(1) Xilinx Spartan 3E FPGA board Manual (Chapter 2, Chapter 3)

2. Printing in the LCD display (3 points)

Write a Verilog module and a constraint file to display "Computer Organization" in the LCD screen of your FPGA Fit. The LCD display has two rows. The word "Computer" should be displayed in the first row and the word "Organization" should be displayed in the second row.

References:
(1) The following Quora post discusses how to display "Hello World" on the LCD display of the Spartan 3E FPGA board:
https://www.quora.com/What-is-the-Verilog-code-to-display-a-character-on-an-LCD-screen-of-the-Spartan-3E-XC3S400-FPGA-kit
(2) Xilinx Spartan 3E FPGA board Manual (Chapter 5)

3a. Comparing two 2-bits binary number (2 points)

(a) Write a Verilog module *eq1* to compare two bits. The output should be 1 if the two bis are equal. Write a Verilog module *eq2* to compare two 2-bit binary numbers. The program should return 1 if the two numbers are equal. Use the module *eq1* to implement the module *eq2*.
(b) Write a Verilog Test Fixture containing at least four different test cases. Use the ISim simulator to simulate the Verilog module.
(c) Write a constraint file to use the 4 sliding switches to input two 2-bit binary numbers. Use one of the LEDs to indicate the output.
(d) Synthesize and implement your Verilog module to generate a configuration file and download the configuration file to the FPGA chip. Move the sliding switches to on and off positions to generate different 2-bit binary numbers as input and check the LED output to ensure that your circuit is correct.

References:
(1) Xilinx Spartan 3E FPGA board Manual (Chapter 2)
(2) Your Verilog Textbook (Chapter 2)

3b. Comparing two 4-bits binary number (3 points)

(a) Write a Verilog module *eq4* to compare two 4-bit binary numbers. The program should return 1 if the two numbers are equal. Use the module *eq2* to implement the module *eq4*.

(b) Write a Verilog Test Fixture containing at least four different test cases. Use the ISim simulator to simulate the Verilog module.

(c) The Xilinx Spartan 3E FPGA board has only four sliding switches. That allows you to input only one 4-bit binary number. To input more data you use push-button switches. You can position the slide switches and press a push button to enter a binary number. You repeat the process with another push button to enter the second number. Update the Verilog module to incorporate the input mechanism using slide switches and push-button switches.

(d) Write a constraint file to use 4 sliding switches and two push buttons to input the two 4-bit binary numbers. Use one of the LEDs to indicate the output.

(e) Synthesize and implement your Verilog module to generate a configuration file and download the configuration file to the FPGA kit. Move the sliding switches to on or off position to generate different 4-bit binary numbers. Use the two push buttons to capture the numbers. Check the LED output to ensure that your circuit is correct.

References:

(1) Xilinx Spartan 3E FPGA board Manual (Chapter 2)

(2) Your Verilog Textbook (Chapter 4, Section 4.5.1 for an example of using slide switch and push-button switch together to input multiple binary numbers)

Assignmnet 2 (10 points)

Lab Session: (during the week of March 27, 2017 - March 31, 2017)
Deadline for submission: 11:59 pm on March 31, 2017

Problems:

1. Adding two 4-bits binary number (1 + 1 + 1 + 2 + 1 = 6 points)

(a) Write a Verilog module *add4* to add two 4-bit binary numbers. The program should return the sum as a four bit number and the carry bit.

(b) Write a Verilog Test Fixture containing at least four different test cases. Use the ISim simulator to simulate the Verilog module.

(c) The Xilinx Spartan 3E FPGA board has only four sliding switches. That allows you to input only one 4-bit binary number. To input more data you use push-button switches. You can position the slide switches and press a push button to enter a binary number. You repeat the process with another push button to enter the second number. Update the Verilog module to incorporate the input mechanism using slide switches and push-button switches.

(d) Write a constraint file to use 4 sliding switches and two push buttons to input the two 4-bit binary numbers. Display the sum as a hexadecimal character on the LCD Display. Use one of the LEDs to indicate the carry output.

(e) Synthesize and implement your Verilog module to generate a configuration file and download the configuration file to the FPGA kit. Move the sliding switches to on or off position to generate different 4-bit binary numbers. Use the two push buttons to capture the numbers. Check the LCD and LED output to ensure that your circuit is correct.

2. Counter with set and reset (2 + 1 + 1 = 4 points)

(a) Write a Verilog module that counts from 0 to 15 repetitively. The counting should happen in every second. The module should have three inputs: (i) *init*, a four bit number that provides the value from which the counting gets started, (ii) *set*, a one bit input that sets the counter to start the counting from the number provided in *init*, and (iii) *reset*, a one bit input that makes the counting start from 0. The module has one four bit output *out* that outputs the current value of the counter. You should use state machine in designing your module.

(b) Write a constraint file to include clock constraints, 4 sliding switches to input the *init* value, two push buttons to generate the *set* and *reset* inputs, and 4 LEDs to display the output value.

(c) Synthesize and implement your Verilog module to generate a configuration file and download the configuration file to the FPGA kit. Check if the LED outputs are changing as desired. Now press the button corresponding to the reset input and see if the counting starts from 0. Next, set the slide switches to a desired value, press the button corresponding to the *set* input and check if the counting starts from the value provided by the sliding switches.

Assignmnet 3 (20 points)

Lab Session: (during the week of April 3, 2017 - April 7, 2017 and April 10, 2017 - April 14, 2017)
Deadline for submission: 11:59 pm on April 14, 2017

Objective: In this lab assignment, you will design a simple processor involving an ALU and a register file.

Problems:

1. 4-bit ALU (2 + 1 + 2 + 1 + 1 = 7 points)

(a) Write a Verilog module *alu4* to perform arithmetic and logical operations on two 4-bit binary numbers. The ALU should support the following operations: and, or, add and sub. It should produce the result of the arithmetic or logical operation as output. Moreover, it should also output three flags CF, ZF and SF to indicate if there is carry generated, if the result is zero and if the result is negative, respectively. For arithmetic operations, assume that the inputs are signed numbers.

(b) Write a Verilog Test Fixture containing at least four different test cases. Use the ISim simulator to simulate the Verilog module.

(c) Update the Verilog module to incorporate the input mechanism using slide switches and push-button switches. Three different push buttons can be used to input the operation (using two slide switches) and the two operands (using four slide switches in each case).

(d) Write a constraint file to use sliding switches and three push buttons to input the two bit opcode and two 4-bit binary numbers and display the result as a hexadecimal character on the LCD Display. Use three of the LEDs to indicate the CF, ZF and SF flags.

(e) Synthesize and implement your Verilog module to generate a configuration file and download the configuration file to the FPGA kit. Move the sliding switches to on or off position to generate different 2-bit numbers for operations and 4-bit binary numbers for operands. Use the push buttons to capture the inputs. Check the LCD screen and LED output to ensure that your ALU is operating correctly.

2. Register File (2 + 2 + 1 + 1 = 6 points)

(a) Write a Verilog module *regfile* to implement a register file that contains 16 registers each of size 4 bits. The register file module should have six inputs: the clock, two read registers (RR1 and RR2), one write register (WR), one write data (WData) port and one write enable (WEnable) port. The module should have two output ports providing the data stored in the two registers given in the two read register input ports. The data provided in the write data port is written to the write register on the positive edge of the clock when the write enable signal is on.

(b) Update the Verilog module to incorporate the input mechanism using slide switches and push-buttons. The four 4-bit inputs RR1, RR2, WR and WData can be obtained using the four slide switches and four push buttons, one for each input. The write enable signal can be generated using the Rotary Push-Button Switch (See page 17 of Spartan-3E FPGA Starter Kit Board User Guide for details).

(c) Write a constraint file to use the sliding switches and four push buttons to input the 4-bit read registers, 4-bit write register and the 4-bit write data. Display the results in two read data ports using eight LEDs.

(d) Synthesize and implement your Verilog module to generate a configuration file and download the configuration file to the FPGA kit. Move the sliding switches to on or off position to generate different 4-bit binary numbers and use the push buttons to capture the register numbers and write data. Check the LED outputs to ensure that the data are written to the registers and read from the registers correctly.

3. A simple processor (3 + 2 + 1 + 1 = 7 points)

(a) Write a Verilog module *processor* that combines modules *alu4* and *regfile* to design a simple processor. The processor has two modes: In one mode, it accepts data to be written to a given register in the register file and in another mode it performs arithmetic and logical operations on the values in the two registers given by the user as input.

The processor accepts a 3-bit opcode, two 4-bits inputs and a one bit input used as register write enable signal. If the opcode is 100 then the processor works in the register write mode. In that case one of the 4-bit

inputs works as the destination register to be written, and the other 4-bit input is the data to be written to that register.

If the opcode is 000, 001, 010 or 011, the processor should perform and, or, add and subtract operations respectively with the help of the arithmetic logic unit. In this case the two 4-bit inputs provide the two registers that contain the operands. In this mode, the processor should output the result and status flags obtained from ALU.

The processor should use an invalid bit as output to indicate the validity of the given opcode. For example, if the input opcode is 110, then the value of invalid bit should be 1.

(b) Update the Verilog module to incorporate the input mechanism using the slide switches and push buttons. The 3-bit opcode and the two 4-bit inputs should be obtained by using the slide switches and three different push buttons. The fourth push button can be used to generate the register write enable signal.

(c) Write a constraint file to use sliding switches and three push buttons to input the 3-bit opcode and two 4-bit binary numbers, and a push button to input the register write enable signal. Include constraints to display the result of the ALU operation as a hexadecimal character on the LCD Display and use three of the LEDs to indicate the CF, ZF and SF flags.

(e) Synthesize and implement your Verilog module to generate a configuration file and download the configuration file to the FPGA kit. First use the opcode 100 to store some values in a few registers. Then try different opcodes for the ALU operations and check the LCD screen and LED output to ensure that your ALU is operating correctly.